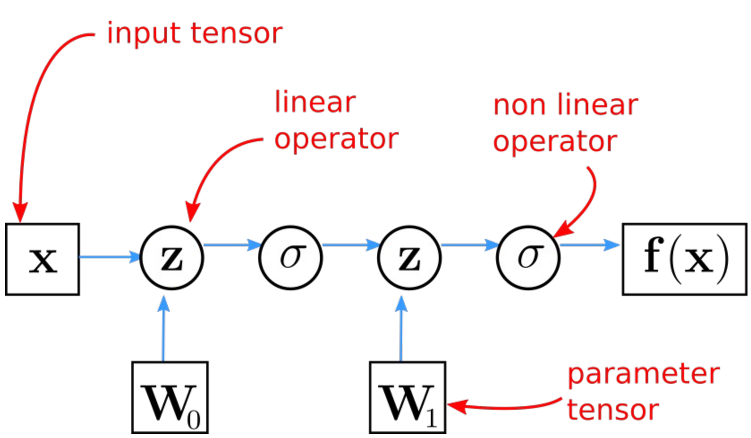
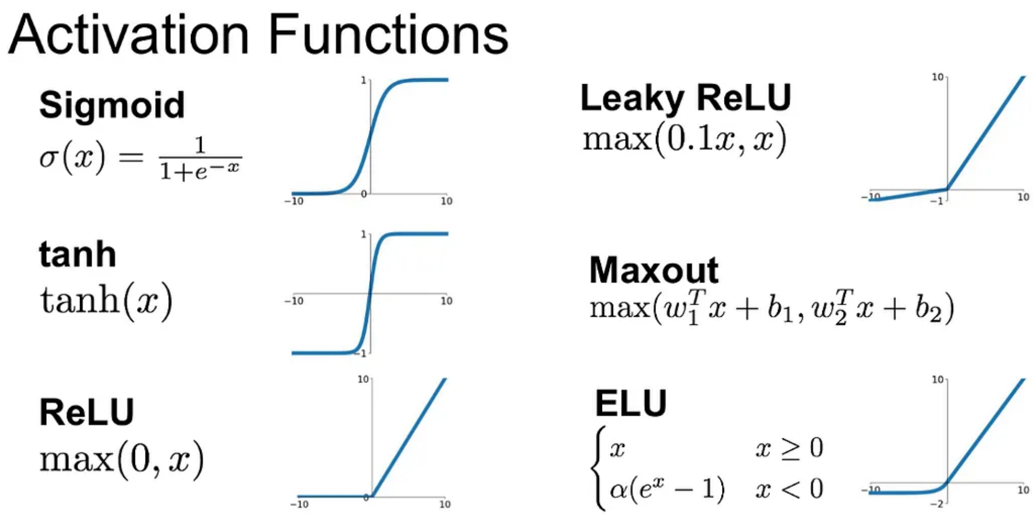
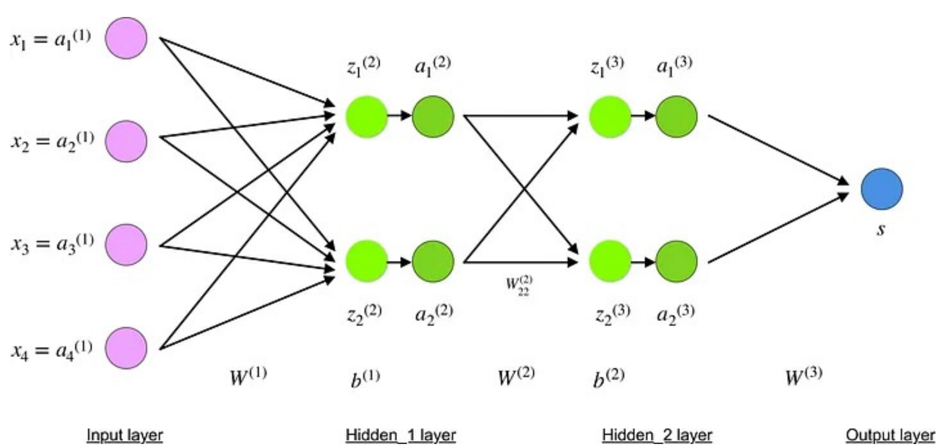
Forward Proporgation



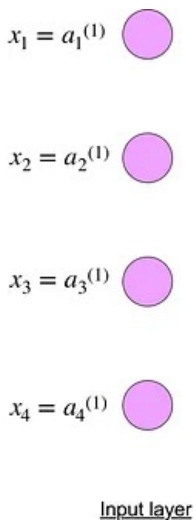
https://m2dsupsdlclass.github.io/lectures-labs/



https://medium.com/@shrutijadon/survey-on-activation-functions-for-deep-learning-9689331ba092

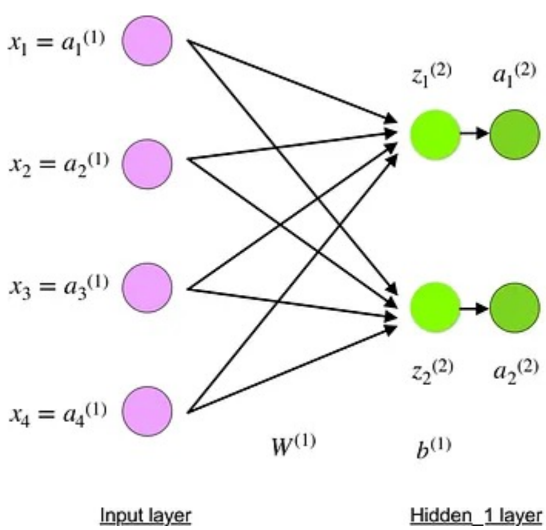


https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd



$$x_i = a_i^{(1)}, i \in 1,2,3,4$$

Equation for input x_i



$$z^{(2)} = W^{(1)}x + b^{(1)}$$

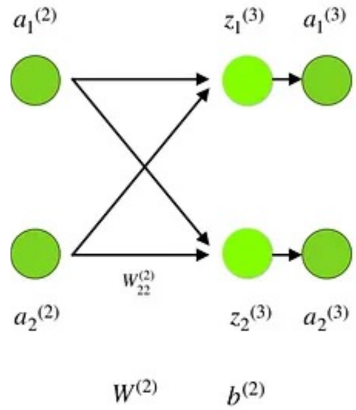
$$a^{(2)} = f(z^{(2)})$$

Equations for z² and a²

$$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$$

$$a^{(3)} = f(z^{(3)})$$

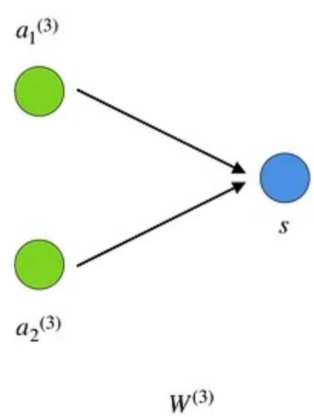
Equations for z^3 and a^3



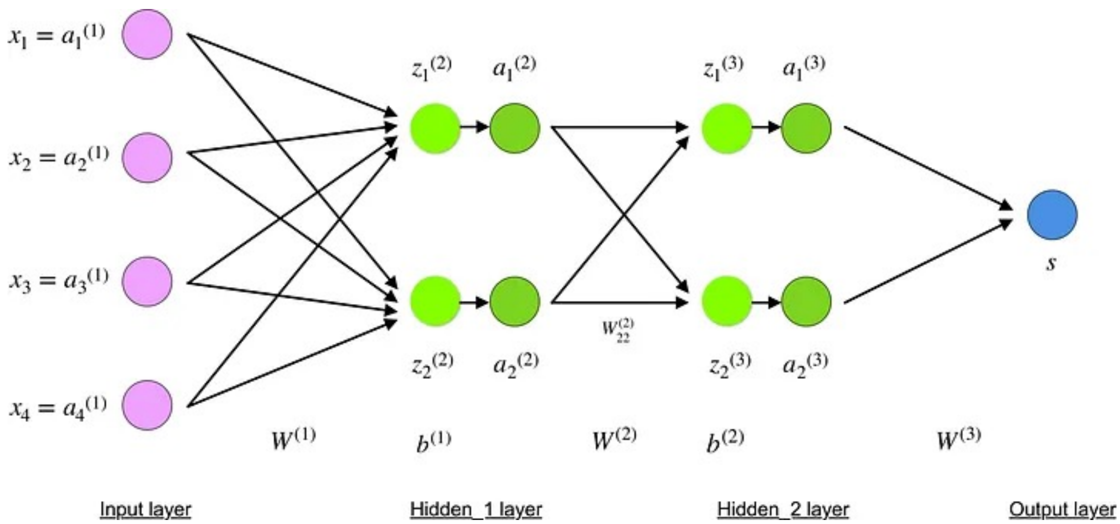
Hidden_2 layer

$$s = W^{(3)}a^{(3)}$$

Equation for output s



Output layer



$x = a^{(1)}$ *Input layer*

$z^{(2)} = W^{(1)}x + b^{(1)}$ *neuron value at $Hidden_1$ layer*

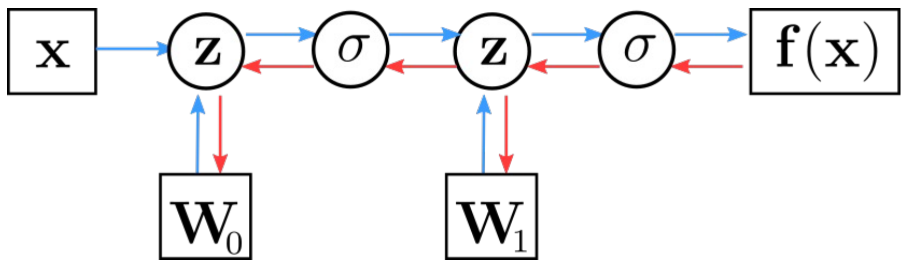
$a^{(2)} = f(z^{(2)})$ *activation value at $Hidden_1$ layer*

$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$ *neuron value at $Hidden_2$ layer*

$a^{(3)} = f(z^{(3)})$ *activation value at $Hidden_2$ layer*

$s = W^{(3)}a^{(3)}$ *Output layer*

Backpropagation

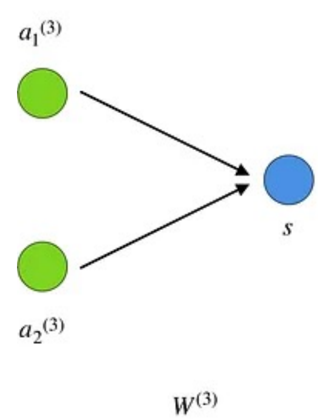


$$s = W^{(3)} a^{(3)}$$

Equation for output s

$$C = cost(s, y)$$

Equation for cost function C



Output layer

The cost function can be MSE, cross-entropy,
or any other cost function

Commons Types of Cost Function

1. **Mean Squared Error (MSE)**:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where y_i is the actual value, $\hat{y}_i$ is the predicted value, and n is the number of samples.

2. **Mean Absolute Error (MAE)**:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

where y_i is the actual value, $\hat{y}_i$ is the predicted value, and n is the number of samples.

3. **Binary Cross-Entropy**:

$$\text{Binary Cross-Entropy} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

where y_i is the actual binary label (0 or 1), $\hat{y}_i$ is the predicted probability, and n is the number of samples.

4. **Categorical Cross-Entropy**:

$$\text{Categorical Cross-Entropy} = -\sum_{i=1}^{n}\sum_{j=1}^{c} y_{ij}\log(\hat{y}_{ij})$$

where y_{ij} is the actual probability (1 or 0) for class j for sample i, $\hat{y}_{ij}$ is the predicted probability for class j for sample i, n is the number of samples, and c is the number of classes.

5. **Hinge Loss** (used for Support Vector Machines):

$$\text{Hinge Loss} = \frac{1}{n}\sum_{i=1}^{n}\max(0, 1 - y_i\hat{y}_i)$$

where y_i is the actual label (-1 or 1), $\hat{y}_i$ is the predicted value, and n is the number of samples.

6. **Huber Loss** (a combination of MSE and MAE):

$$\text{Huber Loss} = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & \text{for } |y_i - \hat{y}_i| \le \delta \\ \delta|y_i - \hat{y}_i| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases}$$

where y_i is the actual value, $\hat{y}_i$ is the predicted value, and δ is a threshold.

7. **Kullback-Leibler Divergence (KL Divergence)**:

$$\text{KL Divergence} = \sum_{i=1}^{n} y_i \log\left(\frac{y_i}{\hat{y}_i}\right)$$

where y_i is the actual probability distribution, $\hat{y}_i$ is the predicted probability distribution.

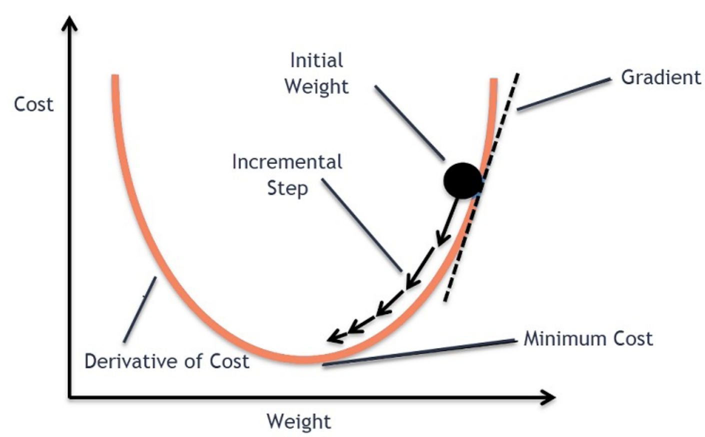
8. **Poisson Loss**:

$$\text{Poisson Loss} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i\log(\hat{y}_i))$$

where y_i is the actual count, $\hat{y}_i$ is the predicted count.

Gradient Descent

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$



Weight gradient

$$\frac{\partial C}{\partial w_{jk}^{l}} = \frac{\partial C}{\partial z_j^{l}} \frac{\partial z_j^{l}}{\partial w_{jk}^{l}} \qquad \textit{chain rule}$$

$$z_j^l = \sum_{k=1}^{m} w_{jk}^{l} a_k^{l-1} + b_j^l \qquad \textit{by definition}$$

$$m \ - \ \textit{number of neurons in } l-1 \textit{ layer}$$

Weight gradient

$$\frac{\partial C}{\partial w_{jk}^{l}} = \frac{\partial C}{\partial z_j^{l}} \boxed{\frac{\partial z_j^{l}}{\partial w_{jk}^{l}}} \qquad \textit{chain rule}$$

$$z_j^l = \sum_{k=1}^{m} w_{jk}^{l} a_k^{l-1} + b_j^l \qquad \textit{by definition}$$

$$m \ - \ \textit{number of neurons in } l-1 \textit{ layer}$$

$$\boxed{\frac{\partial z_j^{l}}{\partial w_{jk}^{l}} = a_k^{l-1}} \qquad \textit{by differentiation (calculating derivative)}$$

$$\frac{\partial C}{\partial w_{jk}^{l}} = \frac{\partial C}{\partial z_j^{l}} \boxed{a_k^{l-1}} \qquad \textit{final value}$$

Constant gradient

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial b_j^l} \qquad \textit{chain rule}$$

Constant gradient

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l}\boxed{\frac{\partial z_j^l}{\partial b_j^l}} \qquad \textit{chain rule}$$

$$\boxed{\frac{\partial z_j^l}{\partial b_j^l} = 1} \qquad \textit{by differentiation (calculating derivative)}$$

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l}\boxed{1} \qquad \textit{final value}$$

Local Gradient

$$\delta_j^l = \frac{\partial C}{\partial z_j^l}$$

Function	Expression	Derivative
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$	$\sigma'(x) = \sigma(x)(1 - \sigma(x))$
Tanh	$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$\tanh'(x) = 1 - \tanh^2(x)$
ReLU	$\text{ReLU}(x) = \max(0, x)$	$\text{ReLU}'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$

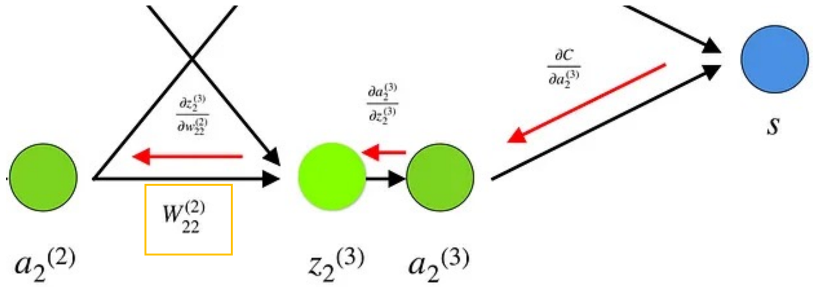
One Update of Parameter

$$while\ (termination\ condition\ not\ met)$$

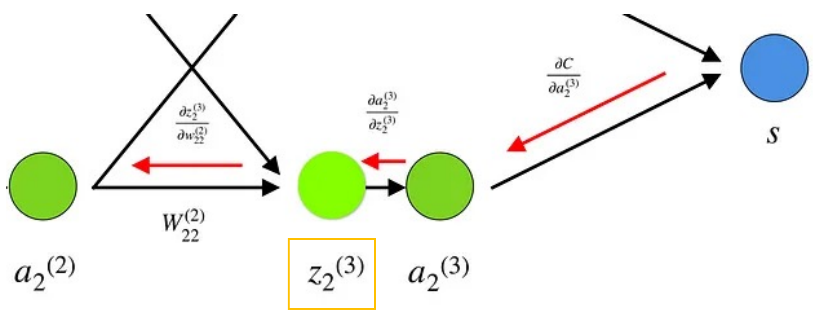
$$w := w - \epsilon \frac{\partial C}{\partial w}$$

$$b := b - \epsilon \frac{\partial C}{\partial b}$$

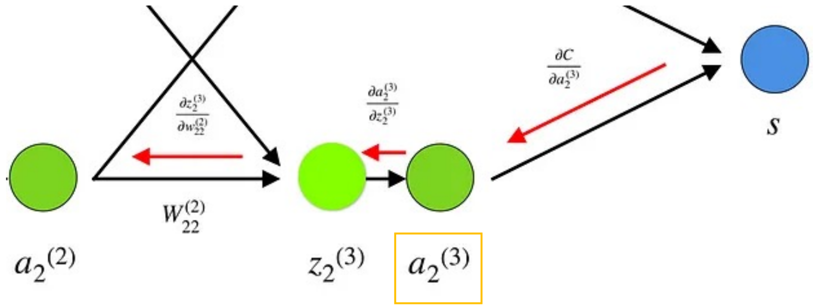
$$end$$



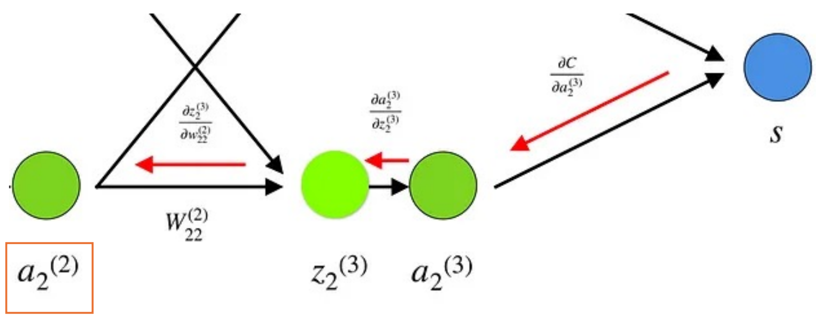
$$\frac{\partial C}{\partial w_{22}^{(2)}}$$



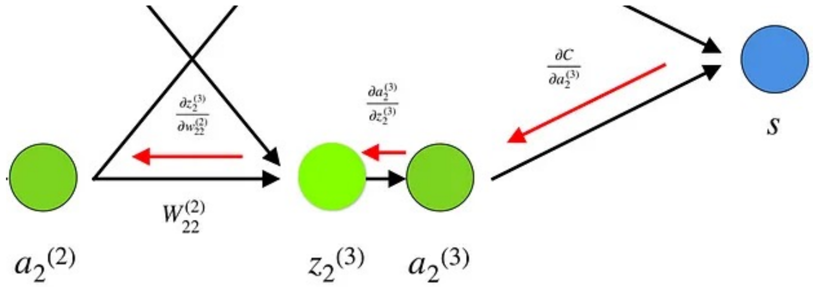
$$\frac{\partial C}{\partial w_{22}^{(2)}} = \frac{\partial C}{\partial z_2^{(3)}} \cdot \frac{\partial z_2^{(3)}}{\partial w_{22}^{(2)}}$$



$$\frac{\partial C}{\partial w_{22}^{(2)}} = \boxed{\frac{\partial C}{\partial z_2^{(3)}}} \cdot \frac{\partial z_2^{(3)}}{\partial w_{22}^{(2)}} = \boxed{\frac{\partial C}{\partial a_2^{(3)}} \cdot \frac{\partial a_2^{(3)}}{\partial z_2^{(3)}}} \cdot a_2^{(2)}$$



$$\frac{\partial C}{\partial w_{22}^{(2)}} = \frac{\partial C}{\partial z_2^{(3)}} \cdot \boxed{\frac{\partial z_2^{(3)}}{\partial w_{22}^{(2)}}} = \frac{\partial C}{\partial a_2^{(3)}} \cdot \frac{\partial a_2^{(3)}}{\partial z_2^{(3)}} \cdot \boxed{a_2^{(2)}}$$



$$\frac{\partial C}{\partial w_{22}^{(2)}} = \frac{\partial C}{\partial z_2^{(3)}} \cdot \frac{\partial z_2^{(3)}}{\partial w_{22}^{(2)}} = \frac{\partial C}{\partial a_2^{(3)}} \cdot \boxed{\frac{\partial a_2^{(3)}}{\partial z_2^{(3)}}} . a_2^{(2)} = \frac{\partial C}{\partial a_2^{(3)}} . \boxed{f^{'}(z_2^{(3)})} . a_2^{(2)}$$

Work Shop

- In `TCC_temple/workshop/starwing_Neural_Net`

- `Linear_Regression.ipynb`
  - `Simple Linear Regression using sklearn`
- `Linear_Regression_nn.ipynb`
  - `Single Layer Linear Regression NN using pytorch`
- `Logistic_Regression.ipynb`
  - `Multilayer Layer Logistic Regression NN using pytorch`

Other Learning Resource

- https://www.kaggle.com/
- https://see.stanford.edu/Course/CS229
- Various Online blogs
- Youtube Video